PTO 04-5191

Japanese Kokai Patent Application
No. Sho 63[1988]-223927

MULTI-PROGRAM SYMBOL DEBUG SYSTEM

Kazuya Hashimoto

UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. AUGUST 2004
TRANSLATED BY THE RALPH MCELROY TRANSLATION COMPANY

JAPANESE PATENT OFFICE
PATENT JOURNAL (A)
KOKAI PATENT APPLICATION NO. SHO 63[1988]-223927

Int. Cl.[4]:	G 06 F 11/28
Sequence Nos. for Office Use:	7343-5B
Filing No.:	Sho 62[1987]-58040
Filing Date:	March 13, 1987
Publication Date:	September 19, 1988
No. of Inventions:	1 (Total of 3 pages)
Examination Request:	Filed

MULTI-PROGRAM SYMBOL DEBUG SYSTEM

[Maruchipuroguramu shinboru debaggu hoshiki]

Inventor: Kazuya Hashimoto

Applicant: NEC Corp.

[There are no amendments to this patent.]

Claim

A symbol debug system characterized by the following facts: in a system that performs debugging of plural programs, there is a symbol table, which at least holds a program identification information that uniquely identifies plural programs, and which also holds symbols defined in the programs corresponding to the program identification information and the property information of the symbols; by assigning the symbol and the program identification information, it is possible to make reference to the symbol table with respect to the assigned program.

Detailed explanation of the invention

Industrial application field

This invention pertains to a programs debug system. Especially, this invention pertains to a symbol debug system.

Prior art

In the prior art, as a tool for performing debugging of programs, a program that executes the programs as the debugger object while making reference to a symbol table for the programs (this program is hereinafter referred to as symbolic debug) is used, and each program has a symbol table.

The symbolic debug in the microprocessor, personal computer, mini-computer, and main frame computer takes only one program as the object. Also, even for debugging with plural programs as object, only one program can be debugged in the symbolic debug.

Problems to be solved by the invention

In order to have plural programs in the memory (for example, operating system (OS) programs and application programs, etc.), in practice, it is necessary to be able to debug plural programs at the same time. Especially, for a system incorporated in a microprocessor (for example, a system including OS having multi-task functions), the debugger object is no longer a single program. Instead, it is the overall system. Consequently, it is necessary to be able to debug the OS program and application program as the structural elements of the system at the same time.

However, in said conventional symbolic debug, only one symbol table is kept for the program, and it is impossible to make identification for the same symbol in different programs. Also, in the case of making reference to the symbol tables of different programs, the debugger user has to issue a control command (hereinafter to be referred to as command) to the debugger, and to perform operation to read the symbol table in the debugger. Consequently, the operation becomes complicated, and this is undesirable.

The purpose of this invention is to solve the aforementioned problems of the prior art by providing a function that can simply assign the symbol table for any program so that symbol debug can be performed with high efficiency.

Means for solving the problems

This invention provides a symbol debug system characterized by the following facts: in a system that performs debugging of plural programs, there is a symbol table, which at least holds program identification information that uniquely identifies plural programs, and which also holds symbols defined in the programs corresponding to the program identification information and the property information of the symbols; by assigning the symbol and the program identification information, it is possible to make reference to the symbol table with respect to the assigned program.

Application examples

In the following, this invention will be explained in more detail with reference to figures. As shown in Figure 1, 101 represents a symbol table. It holds field 104 that contains the program identification information (hereinafter to be referred to as program identification information field), field 107 that contains the symbol values (hereinafter to be referred to as symbol field), and field 108 that contains the property information corresponding to the symbol value (hereinafter to be referred to as symbol property field). The value contained in symbol field 107 and the value contained in symbol property field 108 pair each other, and there are a plurality of them. 102 and 103 are also symbol tables that hold fields just as 101. 105 represents a field identification information field in symbol table 2 102, and 106 represents an identification information field in symbol table 3 103.

Also, 110 represents a symbol assigned in the command with respect to debug, and 109 represents a program identifier assigned at the same time as its symbol. Program identifier 109 holds the value in agreement with the value of the program identification information field of certain symbol table.

In addition, 111 represents the property information with respect to the symbols (hereinafter to be referred to as symbol property information).

In the following, symbol table 1 of 101 will be explained.

First of all, at start of debugging of the program, by reading the symbol table into the debugger by means of the debugger command, the various fields of symbol table 101, that is, program identification information field (104, symbol field 107 and symbol property field 108, are taken as already having values contained in them. Also, symbol tables (102), (103) are also taken as containing the same values. For example, for the value of the program identification information field, the program name corresponding to the symbol table is considered in field 104. In the following explanation, the program name is taken as the value of program identification information field.

By means of program identifier 109 assigned at the same time with symbol 110 in the command with respect to debug, the program identification information field in the plural symbol tables is retrieved, and the program identification information field in agreement with program identifier 109 is determined.

Then, the symbol field in the symbol table containing the determined program identification information field is retrieved, and the symbol field in agreement with symbol 110 in the command is selected, and the value of the symbol property field corresponding to the symbol field is taken as the symbol property information.

By means of the aforementioned constitution, it is possible to perform debugging of plural programs with the symbol present in each program.

Also, said example is merely an application example of this invention. This invention also can be adopted with different methods, that is, the retrieval method of the program identification information with the program identifier as the key, the retrieval method of the symbol field with the symbol as the key, and the method for constructing various fields.

Effects of the invention

As explained in the above, according to this invention, there is a function that can uniquely identify programs, and it can make unique identification of the symbol table of each program, identification of the symbol still can be performed uniquely even when plural identical symbols are present in different symbol tables. Compared with conventional debugging, it is possible to expect higher debug efficiency.

Also, it is possible to make use of the symbol of the program that is currently interrupted in execution by simply knowing the program identifier, that is, the program name, when the program execution is interrupted (breaks) in the debugger, without any operation by the debugger user.

Brief description of the figures

Figure 1 is a diagram illustrating an application example of this invention.

101 Symbol table 1
102 Symbol table 2
103 Symbol table 3
104 Program identification field of symbol table 1
105 Program identification field of symbol table 2
106 Program identification field of symbol table 3
107 Symbol field of symbol table 1
108 Symbol property field of symbol table 1
109 Program identifier
110 Symbol
111 Symbol property information

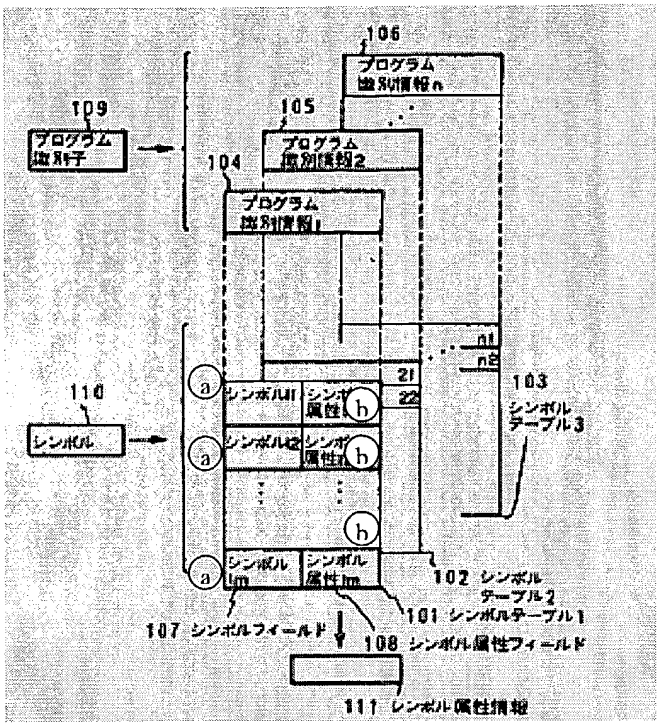

Figure 1

Key:

a	Symbol
b	Symbol property
101	Symbol table 1
102	Symbol table 2
103	Symbol table 3
104	Program identification information 1
105	Program identification information 2
106	Program identification information n
107	Symbol field
108	Symbol property field
109	Program identifier
110	Symbol
111	Symbol property information